УДК 519.683

А. А. Шалыто

АВТОМАТНОЕ ПРОГРАММИРОВАНИЕ*

Что такое автоматно-ориентированное программирование?

В последние годы большое внимание уделяется разработке технологий программирования для встроенных систем и систем реального времени, к которым предъявляются высокие требования по качеству программного обеспечения. Одним из наиболее известных подходов является синхронное программирование [1].

Параллельно с развитием в Европе синхронного программирования в России создается подход к разработке программного обеспечения, названный «автоматно-ориентированным» (или «автоматным») программированием [2–4], который можно рассматривать в качестве разновидности синхронного программирования.

В настоящей работе описываются основные положения автоматно-ориентированного программирования: проектирование, реализация, отладка и документирование программ в части обеспечения корректности их поведения.

В традиционном программировании в последнее время все шире используется понятие «событие», тогда как предлагаемый стиль программирования базируется на понятии «состояние». Понятия «состояние» и «входное воздействие», которое может быть входной переменной или событием, в совокупности образуют «автомат без выхода». Добавляя к ним еще понятие «выходного воздействия», получаем «автомат» (конечный, детерминированный).

Поэтому область программирования, базирующаяся на понятии «автомат», в работе [4] была названа «автоматным программированием», а процесс создания таких программ — «автоматным проектированием программ».

Особенность этого подхода состоит в том, что при его использовании автоматы задаются графами переходов. Для различения однотипных вершин вводится понятие «кодирование состояний». При выборе «многозначного кодирования» с помощью одной переменной можно различить состояния, число которых совпадает с числом возможных значений выбранной переменной. Это позволило ввести в программирование понятия «наблюдаемость» и «управляемость» программ, широко используемые в теории управления.

*При поддержке Министерства образования и науки Российской Федерации в рамках научно-исследовательской работы по теме «Разработка технологии создания программного обеспечения систем управления на основе автоматного подхода».

В рамках предлагаемого подхода программирование выполняется «через состояния», а не «через переменные» (флаги), что позволяет лучше понять и специфицировать задачу и ее составные части. При этом необходимо отметить, что в автоматно-ориентированном программировании проектирование, реализация и отладка проводятся в терминах автоматов. Благодаря этому в рамках предлагаемого подхода от графа переходов к тексту программы можно переходить формально и изоморфно.

В работе [4] было предложено реализовывать автоматы на языках программирования высокого уровня с помощью оператора switch языка C либо его аналогов в других языках программирования. Поэтому предложенная технология была названа «Switch-технология». Ниже будет показано, что для объектно-ориентированного программирования могут быть использованы и другие подходы к реализации автоматов.

В настоящее время эта технология разрабатывается в нескольких вариантах, различающихся как классом решаемых задач, так и типом вычислительных устройств, на которых осуществляется программирование.

Логическое управление

В 1996 г. Российский фонд фундаментальных исследований (РФФИ) в рамках издательского проекта № 96-01-14066 поддержал издание работы [4], в которой предлагаемая технология была изложена применительно к системам логического управления, в которых события отсутствуют, выходные воздействия являются двоичными переменными, а операционная система работает в режиме сканирования. Системы этого класса реализуются обычно на программируемых логических контроллерах, которые имеют относительно небольшой объем памяти, а их программирование выполняется на таких специфических языках, как, например, язык функциональных блоков [5]. В работе [4] предложены методы формального написания программ для этих языков при задании спецификации для разрабатываемого проекта системой взаимосвязанных графов переходов. Показаны преимущества использования языка графов переходов по сравнению с языком «Графсет».

Программирование с явным выделением состояний

В дальнейшем автоматный подход был распространен на событийные системы, которые называются также «реактивными» [6]. В них указанные выше ограничения сняты. Как следует из названия этих систем, в них в качестве входных воздействий используются события, а в качестве выходных – произвольные процедуры.

Для программирования событийных систем с применением автоматов был использован процедурный подход. Поэтому в работе [7] такое программирование было названо «программированием с явным выделением состояний».

При этом выходные воздействия «привязаны» к дугам, петлям или вершинам графов переходов (применяются смешанные автоматы — автоматы Мура–Мили). Это позволяет в компактном виде представлять последовательности действий, которые являются реакциями на соответствующие входные воздействия.

Особенность предлагаемого подхода к программированию этого класса систем состоит в том, что в них повышается централизация логики за счет переноса ее из обработчиков событий и формирования системы взаимосвязанных автоматов, которые вызываются из обработчиков [8]. Автоматы между собой могут взаимодействовать по вложенности, вызываемости и за счет обмена номерами состояний. Последний вид взаимодействия рассматривался также в работе [9], в которой утверждается, что «указанное взаимодействие может оказаться мощным средством при проверке программ».

Система взаимосвязанных автоматов образует системонезависимую часть программы, а функции входных и выходных воздействий, обработчиков событий и т. д. зависят от используемой аппаратно-программной платформы (системы).

Другая важнейшая особенность описываемого подхода состоит в том, что при его применении автоматы используются триедино: при спецификации, при программировании (сохраняются в программном коде) и при отладке. Отладка автоматных программ может выполняться как в графическом режиме (при наличии соответствующих инструментальных средств), так и по протоколам. Отметим, что протоколирование выполняется автоматически по построенной программе и может использоваться для задач большой размерности при сложной логике программы.

При этом каждый построенный протокол может рассматриваться в качестве соответствующего сценария. Отметим, что для «больших» задач невозможно применение диаграмм последовательностей и диаграмм кооперации, входящих в состав языка *UML* [10], так как при использовании этого языка указанные диаграммы предлагается строить вручную на этапе проектирования, в то время как в автоматном программировании протоколы строятся автоматически при выполнении программы.

Протоколы позволяют наблюдать за ходом выполнения программы и демонстрируют тот факт, что автоматы являются не «картинками», а реально действующими сущностями.

Автоматный подход предлагается применять не только при создании системы управления, но и при моделировании объектов управления.

Этот подход был апробирован при разработке ряда систем управления ответственными объектами, в том числе судовыми дизель-генераторами [11]. Система была специфицирована более чем тридцатью взаимодействующими автоматами. Для описания модели дизеля также использовались автоматы. При проектировании на каждый автомат составлялось четыре документа: словесное описание («декларация о намерениях»), схема связей (задает интерфейс автомата, в том числе поясняя на русском языке символы входных и выходных воздействий), граф переходов (с символьными обозначениями входных и выходных воздействий), текст программного модуля, который формально и изоморфно реализует граф переходов (также без использования смысловых идентификаторов и комментариев). Эти документы заменяют самодокументирующиеся программы, содержащие смысловые идентификаторы и комментарии, так как данные средства при сложной логике программы не обеспечивают должного понимания программ и их пригодности для дальнейшей модификации [12]. Эту проблему при сложной логике не решают и самодокументирующиеся графы переходов [10].

Опыт проектирования с применением автоматного подхода подтвердил целесообразность использования протоколов для проверки корректности взаимодействия большого количества автоматов и каждого из них в отдельности.

Объектно-ориентированное программирование с явным выделением состояний

Для решения широкого круга задач весьма эффективен подход, основанный на совместном использовании объектной и автоматной парадигм, который в работе [13] был назван «объектно-ориентированным программированием с явным выделением состояний».

Особенности этого подхода состоят в следующем. Так же как и в машине Тьюринга [14], явно выделены управляющие (автоматные) состояния объекта, число которых значительно меньше числа остальных состояний, например, «вычислительных».

Как и при использовании любого другого подхода, применение предлагаемого подхода связано с множеством эвристик, возвратов назад, уточнений и параллельно выполняемых работ. Однако после завершения создания программы предлагаемый подход может быть сформулирован (по крайней мере, для ее документирования) как «идеальная» технология, фиксирующая принятые решения [15].

1. На основе анализа предметной области выделяются классы и строится диаграмма классов.

2. Для каждого класса разрабатывается словесное описание, по крайней мере, в форме перечня решаемых задач.

3. Для каждого класса создается структурная схема, отражающая его интерфейс и структуру. При этом атрибуты и методы разделены на автоматные и остальные.

4. При наличии в классе нескольких автоматов строится схема их взаимодействия.

5. Для каждого автомата разрабатываются словесное описание, схема связей, граф переходов.

6. Каждый класс реализуется соответствующим модулем программы. Его структура должна быть изоморфна структуре класса, а методы, соответствующие автоматам, реализованы по шаблону, например, приведенному в работе [8].

7. Производится отладка полученной системы, например, путем построения протоколов выполнения, в которых функционирование объектов, содержащих автоматы, описывается в терминах состояний, переходов, событий, входных и выходных воздействий.

8. Выпускается проектная документация, составной частью которой является программная документация.

Описанный подход был использован при создании системы управления «танком» для игры «Robocode» [15]. В отличие от систем управления сотнями других «танков», на этот «танк» подготовлена подробная проектная документация, содержащая, в частности, графы переходов и схемы связей автоматов, реализующих функциональность «танка». Детальные протоколы поведения «танка» позволяют проследить все течение боя. Метод построения протоколов, возможно, является основой для новой концепции построения «черных ящиков».

В описанной технологии автоматы применялись как методы классов. В рамках этой технологии могут быть использованы и другие подходы к объектной реализации автоматов, изложенные, например, в работах [16–18]. Автоматы могут выступать, в частности, как объекты–наследники определенного класса, реализующего базовую функциональность автоматов, обусловленную семантикой Switch-технологии.

Возможно также использование классов, реализующих понятия «состояние» или «группа состояний».

При проектировании автоматных программ может быть использован паттерн «State» или, например, его модификация «StateMachine» [19].

Особенности автоматной реализации параллельных процессов на основе механизма обмена сообщениями рассмотрены в работе [20]. Еще один подход к автоматной реализации параллельных процессов описан в работе [21].

Наличие качественной проектной документации резко упрощает осуществление рефакторинга программы (изменение ее структуры при сохранении функциональности). Последнее подтверждается рефакторингом упомянутой выше системы управления танком, выполненным с целью повышения «объектности» программы [22].

Создано инструментальное средство *UniMod*, которое обеспечивает разработку и выполнение автоматно-ориентированных программ. Этот пакет позволяет использовать *UML*-нотацию при построении диаграмм в рамках Switch-технологии. При этом схемы связей, определяющие интерфейс автоматов, строятся в нотации диаграмм классов языка *UML*, а графы переходов — в *UML*-нотации диаграмм состояний. В состав пакета *UniMod* входит встраиваемый модуль (*plug-in*) для платформы *Eclipse* [23], позволяющий создавать и редактировать *UML*-диаграммы классов и состояний, которые соответствуют схеме связей и графу переходов [24].

Вычислительные алгоритмы

Автоматный подход используется в настоящее время и при реализации вычислительных алгоритмов [25–28]. Так, в частности, показано, что произвольный итеративный алгоритм может быть реализован конструкцией, эквивалентной циклу do-while, телом которого является оператор switch.

На основе автоматов предложен новый подход к построению визуализаторов алгоритмов, используемых на кафедре компьютерных технологий СПбГУ ИТМО при обучении программированию и дискретной математике [29, 30]. Подход позволяет представить логику работы визуализаторов системой взаимосвязанных конечных автоматов. Система состоит из пар автоматов, каждая из которых содержит «прямой» и «обратный» автоматы, обеспечивающие пошаговое выполнение алгоритма вперед и назад соответственно.

На сайте http://is.ifmo.ru введен раздел «Визуализаторы», в котором публикуются визуализаторы, выполняемые в рамках «Движения за открытую проектную документацию».

Движение за открытую проектную документацию

27 ноября 2002 г. на открытии полуфинальных соревнований командного чемпионата мира по программированию *ACM* (Association for Computing Machinery) в Северо-Восточном европейском регионе было объявлено об организации «Движения за открытую проектную документацию» [31]. В рамках этого движения на сайте http://is.ifmo.ru создан раздел «Проекты», в котором размещено более 60 проектов разработки программного обеспечения на основе автоматного подхода. Перечислим некоторые из них:

- автоматная реализация интерактивных сценариев образовательной анимации с использованием Macromedia Flash;
- совместное использование теории построения компиляторов и Switch-технологии;
- скелетная анимация;
- управление различными технологическими процессами и объектами (упрощенная модель цеха холодной прокатки, дизель-генератор, турникет, кодовый замок, светофор, кофеварка, телефон, банкомат, лифт, система безопасности банка и т. д);
- игры («Terrarium», «Robocode», «CodeRally», «Lines», «Bomber», «Однорукий бандит», «Завалинка» и т. д.);
- управление роботами «LEGO Mindstorms»;
- XML-формат для описания внешнего вида видеопроигрывателя (см. на www.crystalplayer.com);
- примеры клиент-серверных приложений;
- построение пользовательских интерфейсов;
- реализация сетевого протокола SMTP.

«Коллекция» проектов пополняется и будет пополняться в дальнейшем.

Заключение

В работе показано, что автоматы в программировании не только служат «для распознавания цепочек символов» [32] и управления «стиральными машинами», являясь одной из математических моделей дискретной математики, но и могут применяться при реализации любых программ, обладающих сложным поведением.

Использование автоматов упрощает формализацию спецификации программы, определяющей ее поведение и играющей «ключевую роль в вопросе сдерживания программных ошибок» [33].

Предлагаемая технология должна ответить на вопрос, поставленный в [34]: «Теорию конечных автоматов мы проходили, но причем здесь программирование?»

Отметим также, что при решении задач логического управления используется стиль программирования «от состояний» по классификации, предложенный в работе [35] (в работе [36] он был переименован в «автоматное программирование» со ссылкой на сайт http://is.ifmo.ru).

Предлагаемая технология является развитием классической теории автоматов [36] и подхода Д. Харела [6, 36], основанного на диаграммах *Statechart*.

Литература

1. Benveniste A., Caspi P., Edwards S. et al. The synchronous languages 12 years later // Proc. of the IEEE. 2003. Vol. 91, № 1. P. 28–35.
2. Шалыто А. А. Алгоритмизация и программирование для систем логического управления и «реактивных» систем: Обзор // Автоматика и телемеханика. 2001. № 1. С. 3–39.
3. Шалыто А. А. Логическое управление. Методы аппаратной и программной реализации алгоритмов. СПб.: Наука, 2000.
4. Шалыто А. А. Switch-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.
5. Шалыто А. А. Реализация алгоритмов логического управления программами на языке функциональных блоков // Промышленные АСУ и контроллеры. 2000. № 4. С. 45–50.
6. Harel D., Politi M. Modeling Reactive Systems with Statecharts. N.Y.: McGraw-Hill, 1998.
7. Шалыто А., Туккель Н. Программирование с явным выделением состояний // Мир ПК. 2001. № 8. С. 116–121; № 9. С. 132–138.
8. Шалыто А. А., Туккель Н. И. Switch-технология – автоматный подход к созданию программного обеспечения «реактивных» систем // Программирование. 2001. № 5. С. 45–62.
9. Дейкстра Э. Взаимодействие последовательных процессов // Языки программирования. М., 1972. С. 9–86.
10. Буч Г., Рамбо Д., Джекобсон А. Язык UML: Руководство пользователя. М.: ДМК, 2000.
11. Шалыто А. А., Туккель Н. И. Проектирование программного обеспечения системы управления дизель-генераторами на основе автоматного подхода // Системы управления и обработки информации. СПб., 2003. Вып. 5. С. 66–82.
12. Безруков Н. Повторный взгляд на «собор» и «базар» // BYTE/Россия. 2000. № 8. С. 60–78.

13. Шалыто А. А., Туккель Н. И. Объектно-ориентированное программирование с явным выделением состояний // Искусственный интеллект – 2002: Материалы междунар. науч.-тех. конф. Таганрог; Донецк, 2002. Т. 1. С. 198–202.

14. Шалыто А., Туккель Н. От тьюрингова программирования к автоматному // Мир ПК. 2002. № 2. С. 144–149.

15. Шалыто А. А., Туккель Н. И. Танки и автоматы // BYTE/Россия. 2003. № 2. С. 69–73.

16. Гуров В. С., Нарвский А. С., Шалыто А. А. Автоматизация проектирования событийных объектно-ориентированных программ с явным выделением состояний // Телематика – 2003: Тр. X Всеросс. науч.-метод. конф. СПб., 2003. Т. 1. С. 282–283.

17. Шопырин Д. Г., Шалыто А. А. Применение класса «STATE» в объектно-ориентированном программировании с явным выделением состояний // Там же. С. 284–285.

18. Корнеев Г. А., Шалыто А. А. Реализация конечных автоматов с использованием объектно-ориентированного программирования // Там же. Т. 2. С. 377–378.

19. Шамгунов Н. Н., Корнеев Г. А., Шалыто А. А. State Machine – новый паттерн объектно-ориентированного проектирования // Информационно-управляющие системы. 2004. № 5. С. 13–25.

20. Гуисов М. И., Кузнецов А. Б., Шалыто А. А. Интеграция механизма обмена сообщениями в Switch-технологию // Проектная документация [Электрон. ресурс]. Режим доступа: http://is.ifmo.ru

21. Любченко В. О бильярде с Microsoft Visual C++ 5.0 // Мир ПК 1998. № 1 [Электрон. ресурс]. Режим доступа: http://www.osp.ru/pcworld/1998/01/202.htm

22. Кузнецов Д., Шалыто А. Система управления танком для игры «Robocode» // Проектная документация [Электрон. ресурс]. Режим доступа: http://is.ifmo.ru

23. [Электрон. ресурс]. Режим доступа: http://www.eclipse.org

24. Гуров В. С., Мазин М. А., Нарвский А. С., Шалыто А. А. UML. Switch-технология. Eclipse // Информационно-управляющие системы. 2004. № 6. С. 12–17.

25. Корнеев Г. А., Шамгунов Н. Н., Шалыто А. А. Обход деревьев на основе автоматного подхода // Компьютерные инструменты в образовании. 2004. № 3. С. 32–37.

26. Шалыто А., Туккель Н., Шамгунов Н. Задача о ходе коня // Мир ПК. 2003. № 1. С. 152–155.

27. Шалыто А. А., Туккель Н. И. Преобразование итеративных алгоритмов в автоматные // Программирование. 2002. № 5. С. 12–26.

28. Туккель Н. И., Шалыто А. А., Шамгунов Н. Н. Реализация рекурсивных алгоритмов на основе автоматного подхода // Телекоммуникации и информатизация образования. 2002. № 5. С. 72–99.

29. Корнеев Г. А., Казаков М. А., Шалыто А. А. Построение логики работы визуализаторов алгоритмов на основе автоматного подхода // Телематика – 2003: Тр. X Всеросс. науч.-метод. конф. СПб., 2003. Т. 2.

30. Казаков М. А., Шалыто А. А. Использование автоматного программирования для реализации визуализаторов // Компьютерные инструменты в образовании. 2004. № 2. С. 19–33.

31. Шалыто А. А. Новая инициатива в программировании – «Движение за открытую проектную документацию» // Мир ПК. 2003. № 9. С. 52–56.

32. Хопкрофт Д., Мотвани Р., Ульман Д. Введение в теорию автоматов, языков и вычислений. М.: Вильямс, 2002.

33. Аллен Э. Типичные ошибки проектирования. СПб.: Питер, 2003.

34. Чижов А. [Электрон. ресурс]. Режим доступа: chizh@irk.ru

35. Непейвода Н. Н., Скопин И. Н. Основания программирования. Ижевск: Науч.-издат. центр «Регулярная и хаотическая динамика», 2003.

36. Непейвода Н. Н. Стили и методы программирования. М.: Интернет-Университет информационных технологий – ИНТУИТ.РУ, 2005.

37. Harel D. Statecharts: A visual formalism for complex systems // Science of Computer Programming. 1987. Vol. 8. P. 231–274.